

Istituto Comprensivo "DANTE ALIGHIERI"  
Via Giustiniano, 7 - 34133 TRIESTE



**IL LOGO:**  
**un linguaggio di programmazione per apprendere**  
(programma di riferimento Berkeley Logo)

*Anno scolastico 2005/2006*

*prof. Maria Teresa Mecchia*  
e-mail:mecchia@tin.it

---

Copyright © 2005 MT Mecchia

Tutte le pagine di questa dispensa sono soggette al diritto di copyright e date in licenza d' uso tramite la GNU Free Documentation Licence ( Licenza per la Documentazione Libera), della Free Software Foundation, che in sintesi prescrive:  
Chiunque può copiare e ridistribuire liberamente questo testo, purchè ne preservi la dichiarazione di copyright e di assoggettamento alla Licenza.  
Questa limitazione ha l' unico scopo di non sottrarre ad altri lo stesso diritto di copia.

# **IL LOGO: un linguaggio di programmazione per apprendere**

**Se ascolto dimentico  
Se vedo ricordo  
Se faccio capisco**

Logo e' un linguaggio di programmazione, per alcuni versi di facile apprendimento, per altri davvero complesso: è un dialetto del LISP (linguaggio elettivo assieme al Prolog, dell'approccio all'intelligenza artificiale). Per meglio dire, è una versione procedurale e insieme funzionale del Lisp, che è solo funzionale.

Di solito si pensa al Logo come a un linguaggio elementare, ma si tratta di una splendida palestra per esercizi di programmazione molto sofisticati: qualcuno lo propone come modello base per apprendere la Computer Science, o come strumento per l'esplorazione della matematica avanzata. Offre opportunità in più rispetto al Pascal, altro linguaggio pensato per la didattica della programmazione.

Con gli ambienti messi a disposizione dal Logo si possono insegnare i fondamenti della programmazione di automi a bambini e ragazzi della fascia dell'obbligo in una modalità gioiosa e spensierata che fa impallidire i più prestigiosi giochi.

Il Logo insegna a programmare in maniera modulare, insegna al programmatore a frammentare il suo progetto in molti sottoprogetti ricomponibili in vario modo, sia per raggiungere l'idea iniziale, sia per realizzare poi altri progetti trovando i "mattoni" già pronti: un poco come si potrebbe fare con il Lego se, una volta realizzata una struttura di pochi mattoncini, la si potesse reduplicare all'infinito.

Logo e Lego hanno molte affinità'.

Con il Lego si possono costruire degli automi comandabili con il Logo (LogoLego)

La capacità di strutturare modularmente il proprio pensiero e' utile sia in ambito informatico che nell'apprendimento in generale: fu Papert, informatico del MIT che, seguendo le impostazioni pedagogiche di Piage' realizzò il Logo che venne e viene usato in moltissime scuole nel mondo .

## **Piccoli consigli iniziali**

Se vuoi che il LOGO esegua i tuoi comandi, devi porre molta attenzione nello scrivere gli ordini correttamente e nel mettere gli spazi quando necessario.

Se devi eseguire un disegno che tu hai definito, e insegnato al Logo con il nome di *cassa*, e invii il comando di eseguire *cassa*, lui ti risponderà che non sa come fare; sul video apparirà la scritta

```
I DON'T KNOW HOW TO cassa
```

Lo stesso accade se dimentichi parentesi, virgolette o altro: insomma la sintassi del linguaggio LOGO richiede molta attenzione e, anche se usi un linguaggio più vicino all'uomo, devi sempre ricordare che per un computer tutto, anche uno spazio bianco, ha un significato.

Quando avvii il programma ti troverai davanti ad una finestra divisa in due parti: una parte nera nella quale visualizzerai il risultato dei tuoi comandi, un editor box in basso, generalmente bianco, nel quale scriverai i comandi che vuoi dare alla tartaruga. Generalmente l'editor box è di 4 righe.

Se invece vedi una finestra che contiene solo testo, al primo comando di grafica comparirà la parte nera con la tartaruga.

## **I COMANDI PRINCIPALI**

I comandi alla tartaruga possono venir dati in forma estesa o abbreviata, e indifferentemente in caratteri maiuscoli o minuscoli.

La tartaruga una volta era un piccolo robot che si muove a comando su ruote: adesso la vedi sullo schermo come un triangolo isoscele. I due lati eguali formano una freccia, la direzione dove "sta guardando" la tartaruga. Il terzo lato è la "coda": possiamo pensare di guardare dall'alto un motoscafo con una prua appuntita e la "poppa" dritta.

### **Comandi di spostamento:**

devono essere seguiti da uno spazio e da un numero

**FD** (FORWARD) *distanza*

fa avanzare la tartaruga di *distanza* passi.

Es: FD 50

**BK** (BACK) *distanza*

fa indietreggiare la tartaruga , in retromarcia.

Es. BK 50

**RT** (RIGHT) *angolo*

fa cambiare la direzione della tartaruga, ruotando a destra dell' angolo dato (in gradi!!).

Es; RT 30

**LT** (LEFT) *angolo*

fa cambiare la direzione della tartaruga ruotando a sinistra dell'

angolo dato.

Es: LT 45

### **Comandi sullo stato della tartaruga**

**HT** (HIDETURTLE)

nasconde sullo schermo la tartaruga che pero' scrive ancora.

**ST** (SHOWTURTLE)

fa vedere la tartaruga

**HOME**

in qualunque posto si trovi la tartaruga torna a casa (al centro dello schermo, detto tana) ma non cancella le traccie che ha gia' fatto. Attenzione: se la penna è giù, nel tornare, lascia traccia visibile del suo movimento.

### **Comandi sullo stato dello schermo**

**CS** (CLEARSCREEN)

pulisce lo schermo (la parte grafica) e riporta la tartaruga alla tana

**CT** (CLEARTEXT)

cancella il testo dallo schermo (la parte scritta)

**TS** (TEXTSCREEN) tutto lo schermo viene utilizzato per la modalità testuale e si vede solo il testo.

**SS** (SPLITSCREEN)

parte dello schermo viene usato per la modalità grafica, rimangono per il testo alcune righe in basso

**FS** (FULLSCREEN)

tutto lo schermo viene usato per la modalità grafica. In UCBlago per Windows, però, rimangono sempre alcune righe in basso.

### **I comandi della penna**

La tartaruga robot porta una penna esattamente al centro; per la tartaruga sullo schermo, la penna è a metà della "linea di poppa". Puoi scegliere in che modo deve trovarsi la penna della tartaruga, se deve scrivere, cancellare o altro:

**PD** (PENDOWN)

la penna è giù e "appoggia" sulla carta normalmente, è lo stato in cui si trova la penna quando si inizia a lavorare in LOGO.

**PPT** (PENPAINT)

La penna scrive normalmente, anche in questo stato si trova la penna quando si inizia a lavorare in LOGO.

**PU** (PENUP)

la penna viene "alzata": non scrive, non cancella e non inverte

**PE** (PENERASE)

la penna cancella tutto ciò su cui passa: se era su, viene messa giù e cancella.

**PX** (PENREVERSE)

la penna viene messa giù e cancella se passa su parti disegnate, disegna se passa su parti non disegnate.

Attenzione; se si vuole usare la penna come una gomma bisogna scrivere PE, da allora tutti i comandi faranno tracce che cancellano, per far ritornare a scrivere la tartaruga

bisognerà scrivere il comando PENPAINT (abbreviato PPT).

### DEFINIAMO UNA PROCEDURA

TO *nome della procedura*

serve per aprire una procedura, e' seguito dal nome che si vuol dare alla procedura, quindi si va a capo e si elencano i comandi, la procedura va sempre chiusa andando a capo e mettendo un END.

Se la procedura non viene chiusa, tutto quello che si scrive continua ad esser messo nella procedura.

Per abbandonare una procedura senza terminarla, la si deve bloccare, premendo contemporaneamente i tasti Ctrl e Q.

Es:

```
TO QUADRATO
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
END
```

**Note:** l'ultimo comando, RT 90, non serve per finire di disegnare il quadrato: serve a riportare la tartaruga nella stessa direzione che aveva all' inizio. Dopo aver eseguito QUADRATO la tartaruga si trova nello stesso punto e con la stessa orientazione di partenza.

QUADRATO e' perciò una "**procedura trasparente**", che puo' venir aggiunta o tolta da un'altra procedura senza particolari precauzioni.

E' una buona abitudine scrivere sempre procedure trasparenti.

I comandi possono esser scritti anche tutti di fila, e' obbligatorio andare a capo solo dopo il nome della procedura e prima dell'END.

```
REPEAT numero[Lista di istruzioni]
```

serve per non ripetere tante volte comandi o procedure uguali.

Es:

```
TO QUADRATO  
REPEAT 4[FD 50 RT 90]  
END
```

Ogni procedura del LOGO, una volta definita, diventa un mattone con cui costruire altre parti piu' complesse.

Se io volessi disegnare un rosone facendo ruotare il QUADRATO che ho definito ( ma potrebbe essere anche TRIANGOLO o RETTANGOLO o PLUTO) basterebbe indicarne il nome:

```
TO ROSONE  
REPEAT 36[QUADRATO RT 10]  
END
```

ROSONE

### **GUARDIAMO DENTRO UNA PROCEDURA**

```
EDIT "nomedellaprocedura  
(le virgolette iniziali sono necessarie!)
```

Con questo comando si puo' andare su un foglio particolare chiamato EDITOR nel quale e' possibile correggere una procedura (qui il cursore si sposta in su' e in giu'), cambiarle nome, modificarla.

Per uscire dal foglio dell' EDITOR, dipende da che editor si sta usando. Sotto Windows, UCBlago usa il blocco note. Basta chiudere la finestra perche' il blocco note chieda se si vogliono tenere per buone o abbandonare le modifiche apportate.

Se si vuole invece abbandonare l' Editor e tornare alla situazione di partenza, lasciando tutto invariato si premono contemporaneamente i tasti Ctrl e Scroll (Bloc).

```
PO "nomedellaprocedura sta per Print Out
```

stampa sullo schermo la procedura, ma non e' possibile modificarla.

### **GUARDIAMO NELLA MEMORIA VOLATILE** (workspace o spazio di lavoro)

**POTS** sta per Print Out TitleS

fa comparire sullo schermo l'elenco di tutte le procedure presenti in memoria-volatile, cioe' l'elenco di tutte le procedure che ho costruito fino a quel momento e di tutte quelle che ho eventualmente caricato dal dischetto (vedi comando LOAD).

**POPS** sta per Print Out ProcedureS

fa comparire sullo schermo l'elenco ed il contenuto di tutte le procedure presenti in memoria volatile.

**PO** [lista di procedure]

fa comparire sullo schermo l'elenco ed i contenuti delle sole procedure indicate all'interno delle parentesi.

### **COME CANCELLARE DALLA MEMORIA VOLATILE**

**ERASE** "nomedellaprocedura"      **ER** "nome....."

cancella ,dalla memoria volatile, la procedura indicata.

**ERPS** sta per ERase ProcedureS

cancella tutte le procedure presenti in memoria volatile.

### **E ORA.....METTIAMO IN SALVO IL NOSTRO LAVORO**

Ogni volta che esci dal LOGO oppure spegni il computer e non hai operato un "salvataggio" del tuo lavoro, tutto cio' che hai prodotto e che e' registrato sulla memoria volatile della macchina, viene perso, cancellato.

Occorre quindi salvare tutto da qualche parte: se non lo istruisci diversamente, UCBlogo sotto Windows salva tutto nella cartella "Documenti" dell' utente che ha aperto la macchina.

*(per istruire UCBlogo a salvare altrove, leggi in fondo)*

**SAVE "nomediunfileinventato**

Con questo comando tutto quello che c'e' in memoria volatile viene salvato in uno spazio del disco o file nella cartella Documenti, che porterà il nome da te dato {nomeinventato}).

Es:

SAVE "LUIGI

aspetta finché il logo ti propone nuovamente il punto di domanda. Non ci vorrà molto, all' inizio.

Ora il tuo lavoro è stato registrato sul disco, nella cartella Documenti, nel file LUIGI; puoi spegnere tranquillamente il computer, la prossima volta che lo riaccenderai potrai ricaricare il tuo lavoro nel logo con il comando

LOAD "LUIGI

*(Però se pensi che la prossima volta lavorerai su un PC diverso devi salvare LUIGI su dischetto o pendrive: vedi nota in fondo su come si fa).*

UCBlogo non ha un comando che fa comparire la lista di tutti i file presenti nella cartella di lavoro (Documenti, se non gli dici altrimenti). Se hai dimenticato in che file hai salvato delle procedure, devi guardare nella directory Documenti con Esplora Risorse. Tutti i nomi dei file sono formati da una sequenza di caratteri che può terminare con un' estensione (le lettere che seguono il punto): conviene usare l' estensione per distinguere i file Logo dagli altri che possono essere nella cartella Documenti: usa .lg o .lgo come estensione:

SAVE "luigi.lg  
LOAD "luigi.lg

**LOAD "nomedelfile**

carica il contenuto del file chiamato in memoria volatile.

Da questo momento in poi avrai, in memoria volatile, tutto quello che c'era prima piu' tutto quello che hai caricato. Attenzione ; se, in memoria volatile, c'era una procedura con un nome uguale ad una procedura caricata, ma con significato diverso, il LOGO sostituisce la prima con l'ultima caricata. Per verificare ora quello che c'e' in memoria volatile ti conviene fare POTS.

Es: LOAD "PIPPO  
carica il file di nome PIPPO ed estensione LF.

**ERASEFILE** "*nomedelfile*

comando da usarsi con attenzione!  
Cancella definitivamente dal dischetto il file indicato.  
Per l'estensione valgono le regole di sopra.

**EDITFILE** "*nomedelfile*

Questo comando consente di entrare nel foglio dell' editor, su cui si troverá il contenuto del file, e consente di fare aggiunte e modifiche.

Valgono le regole di EDIT.

E' un comando da usarsi con attenzione!

### **Aggiungiamo una procedura ad un file già esistente**

Per aggiungere una procedura ad un file esistente si puo' :  
1) caricare il file in memoria  
2) salvare tutte le procedure presenti nella memoria volatile. Questo file conterra' ora le procedure nuove e le vecchie.

Es:

LOAD "PIPPO  
carico il file PIPPO, faccio

POTS  
per verificare che ci siano tutte le procedure che mi interessano, salvo di nuovo tutto su PIPPO

SAVE "PIPPO

E' un sistema un po' pericoloso, perche' si rischia di perdere tutto il contenuto di un file se si commettono errori o se qualcuno ti spegne la macchina mentre lavori. Conviene allora farsi una copia del file, per sicurezza, e poi cancellare.

Per fare una copia del file *dal logo* si usano questi comandi:

```
DRIBBLE "PIPPO.OLD POFILE "PIPPO NODRIBBLE
```

In questo modo copi il contenuto del file PIPPO in un nuovo file con lo stesso nome, ma con estensione diversa: PIPPO.OLD.

Quando avrai finito di fare le modifiche e le avrai salvate di nuovo su PIPPO.LF potrai anche cancellare PIPPO.OLD

```
ERASEFILE "PIPPO.OLD
```

Ma per evitare i rischi ti conviene prima fare una copia del vecchio file da Esplora Risorse, e poi rinominarlo: per fare un po' d' ordine e non rischiare di perdere i file ti suggerisco di rinominare i file includendo nel nome la data di oggi, per esempio:

```
PIPPO.2006.03.28.lg
```

Ci sono anche altri sistemi che prevedono l'impacchettamento delle procedure.

### **INIZIAMO A LAVORARE CON LE VARIABILI**

Fino ad ora abbiamo definito procedure con misure fisse: il lato del QUADRATO usato come esempio era di 50 passi; se volessimo disegnare un QUADRATO con un lato diverso dovremmo riscrivere tutta la procedura.

Si puo' definire la procedura QUADRATO dicendo al LOGO che la misura del lato verra' data ogni volta che viene chiamata la procedura, il che fa diventare il lato una variabile. Per dire al LOGO che seguira' una variabile si usano i due punti.

```
TO QUADRATO :LATO           oppure           TO QUADRATO :LATO
  FD :LATO                   REPEAT 4[FD :LATO RT
90]
```

```

RT 90                                END
FD :LATO
RT 90
FD :LATO
RT 90
FD :LATO
RT 90
END

```

Posso far variare anche l'angolo, e allora non avro' piu' un quadrato ma, se metto l'angolo di rotazione giusto, un poligono.

Per dare l'angolo di rotazione giusto ci serviremo del Teorema della tartaruga

#### **TEOREMA DELLA TARTARUGA:**

**La tartaruga deve ruotare sempre di 360 gradi (o multipli) per ritornare nella direzione di partenza.**

Se voglio disegnare un triangolo regolare faro' fare alla tartaruga una rotazione completa in tre tappe uguali: i tre angoli di rotazione. L'angolo di rotazione sara'  $360/3$ .

*N.B. l'angolo di rotazione corrisponde all'angolo complementare esterno del triangolo.*

Se voglio disegnare un pentagono regolare l'angolo di rotazione sara'  $360/5$ .

Se voglio disegnare un poligono regolare di n lati, l'angolo di rotazione sara'  $360/n$ .

Scriviamo la procedura di un poligono con variabile la misura del lato e del numero di lati

```

TO POLIGONO :LATO :N                N sta per numero dei
lati
REPEAT :N [FD :LATO RT 360/:N]
END

```

Es:

```

se voglio un quadrato di lato 60
POLIGONO 60 4

```

se voglio un dodecagono di lato 10  
POLIGONO 10 12

### **LE PROCEDURE RICORSIVE e due nuovi comandi IF e STOP**

La potenza del LOGO sta nel fatto che tu puoi individuare un progetto, suddividerlo in tante procedure tra loro separate, ognuna con un nome proprio, che puo' chiamare un'altra procedura od esserne a sua volta chiamata.

Una procedura ricorsiva chiama se stessa.

Es:  
TO QUA  
FD 50 RT 90  
QUA  
END

questa procedura ricomincia ogni volta da capo ripartendo da dove si e' posizionata la tartaruga. Disegnera' un quadrato e ci girera' sempre intorno.

Per bloccare una procedura ricorsiva "senza condizione di terminazione" dovro' premere una combinazione di tasti speciali: con UCBlogo su Windows dovro' premere contemporaneamente i tasti Ctrl e Q. (quit)

Una procedura si puo' anche mettere in pausa, premendo contemporaneamente i tasti Ctrl e W (wait): per ripartire dopo una pausa, serve il comando

**CO** oppure **CONTINUE**

Quest' altro esempio e' piu' interessante, ma dobbiamo rallentarlo un po' dopo ogni chiamata perche' altrimenti non si riesce assolutamente a seguire la tartaruga, troppo veloce: il comando

**WAIT** 15

ferma l' esecuzione per 15/60 (cioe' un quarto) di secondo ogni volta.

Es:  
TO SPIRALE :L  
FD :L RT 90

```
WAIT 15
SPIRALE :L+2
END
```

si puo' variare anche l'angolo di rotazione, o di volta in volta, o facendolo diventare una variabile:

```
TO SPIRALE :L :ANG
FD :L RT :ANG
  WAIT 15
SPIRALE :L+2 :ANG
END
```

Anche qui la procedura non si ferma mai, bisogna fermarla con Ctrl - Q (oppure con Ctr - W e poi CONTINUE) oppure introducendo nella procedura una condizione di blocco. Decidiamo, per esempio, di fermare la procedura quando il lato diventa maggiore di 80.

Per far questo ci serviremo di due nuovi comandi: IF e STOP

### **IF**

serve per verificare se l'affermazione che segue e' vera. Se e' vera allora il LOGO esegue la lista di istruzioni messe di seguito tra parentesi quadre. Se e' falsa esegue o una seconda lista di istruzioni messe tra parentesi quadre, oppure, se la seconda lista non viene messa, continua indisturbato ad eseguire la procedura. Noi useremo l'IF quasi sempre nel primo caso: se si verifica la condizione posta, il LOGO esegue quanto c'e' di seguito tra parentesi quadre, se non si verifica il LOGO prosegue.

### **STOP**

e' un comando che serve a bloccare la procedura che il LOGO sta eseguendo. Il LOGO non ferma quando viene eseguita un'istruzione STOP: si limita a terminare l'esecuzione della procedura che contiene, esattamente come avviene quando viene incontrata la fine di una procedura (comando END).

Torniamo alla nostra spirale e blocchiamola .

```
TO SPIRALE :L
IF :L > 80 [STOP]
```

```
FD :L RT 90
SPIRALE :L + 2
END
```

### IL CAMPO DI AZIONE DELLA TARTARUGA

La tartaruga ha sempre due caratteristiche :la posizione e l'orientazione. L'orientazione puo' venir espressa in gradi, esattamente come in una bussola, come puoi vedere dallo schema che segue.

Si puo' orientare la tartaruga nella posizione voluta a prescindere da dove si trovava, usando il comando SETHEADING.

I	0	I
I	Nord	I
I		I
I		I
I		I
I 270		90 I
IOvest		Est I
I		I
I		I
I		I
I	Sud	I
I	180	I
I		I

**SETHEADING** *angolo* oppure **SETH** *angolo*

i valori dell' angolo possono essere compresi tra -9999 e 9999.

I valori positivi procedono in senso orario a partire dal nord.

Es:

CS

SETH 90

RT 90

SETH 90 la tartaruga si posiziona sull'Est.

Notate la differenza tra il comando D e il comando SETH : D 90

fa ruotare ogni volta la tartaruga di 90 verso destra, SETH 90

riposiziona sempre la tartaruga in direzione est.

## HEADIG

e' un' operazione che da', come risultato (l'output), l'orientazione della tartaruga.

se vuoi chiedere al LOGO che orientamento ha la tartaruga e scrivi HEADIG il LOGO ti rispondera':

```
I DON'T KNOW WHAT TO DO WITH
```

e scrivera' un numero, per es. 90.

Cio' accade perche' non gli e' stato detto cosa fare dell'output di HEADIG.

Dopo un operazione occorre dare un comando, ad esempio facciamo stampare il risultato usando il comando **PRINT**:

```
PRINT HEADIG
```

90 e' la risposta del LOGO

**PRINT** *oggettodellastampa* oppure **PR** *oggettodellastampa*

serve per stampare sullo schermo cio' che ci interessa.

Prova:

se, dopo PRINT, segue una operazione	PRINT 3+5
se, dopo PRINT, segue un nome	PRINT "MARIA
se, dopo PRINT, segue una lista	PRINT [EVA MARIA LUCA]

## POS

e' una operazione che restituisce una lista composta da due elementi: il primo e' la coordinata x (espressa in passi tartaruga) della posizione in cui si trova la tartaruga (o, per essere piu' precisi, la punta della sua penna); il secondo e' la coordinata y. Per esempio, se la tartaruga si trova nella tana, al centro, la posizione sara' [0 0].

Essendo un' operazione deve essere preceduta da un comando, ad es. il comando di stampa.

PRINT POS

[0 0] e' la risposta del LOGO

La tartaruga puo' muoversi sullo schermo considerandolo un piano cartesiano i cui assi hanno origine nel centro (nella tana) : l'asse verticale , asse delle y, ha i valori positivi crescenti verso l'alto e i valori negativi decrescenti verso il basso; l'asse l'asse orrizontale, asse delle x, ha i valori positivi crescenti verso destra (est) e i valori negativi decrescenti verso sinistra (ovest).

La tartaruga puo' esser individuata sullo schermo da una coppia di coordinate cartesiane, ugualmente, puo' muoversi sullo schermo raggiungendo punti individuati da coordinate cartesiane.

#### **XCOR**

e' una operazione che calcola il valore della x del punto in cui si trova la tararuga.

#### **YCOR**

e' una operazione che calcola il valore della y del punto in cui si trova la tartaruga.

#### **SETX** numero

e' un comando che sposta la tartaruga orrizontalmente nel punto di coordinata x dato con SETX.

#### **SETY** numero

e' un comando che sposta la tartaruga verticalmente nel punto di coordinata y dato con SETY.

#### **SETPOS** [coordinataX coordinataY]

e' un comando che sposta la tartaruga nel punto di coordinate date.

Definiamo la procedura AQUADRATO usando questo sistema

TO AQUADRATO

```
SETPOS [0 50]
SETPOS [50 50]
SETPOS [50 0]
SETPOS [0 0]
END
```

il quadrato cosi' definito viene stampato nel quadrante positivo, in alto a destra, dello schermo.

Notate la differenza tra questo quadrato e quello definito con i comandi A 50 D 90 : AQUADRATO non e' trasferibile, perche' fissato nei punti di coordinate date.

```
AQUADRATO RT 30 AQUADRATO
```

il LOGO disegna due quadrati, uno sovrapposto all'altro, e la tartaruga resta orientata 30 gradi a destra dell'orientazione che aveva prima.

### **COLORIAMO I NOSTRI DISEGNI**

UCBlogo puo' rendere fino a 281 mila miliardi di colori, usando una lista di tre numeri "RGB" compresi fra 0 e 65535 che descrive le componenti di luminosita' Rosso, Verde (Green) e Blu: cosi' ad esempio [0 0 0] rappresenta il nero, [65535 65535 65535] il bianco, [30000 30000 30000] un grigio intermedio; il rosso puro e' [65535 0 0], il verde puro [0 65535 0] e il blu puro [0 0 65535].

Per semplificare le cose UCBlogo permette di definire delle tavolozze, o PALETTE, con un numero limitato di colori ben definiti. Una tavolozza e' gia' definita all'apertura e consiste di 16 colori, ognuno rappresentato da un solo numero:

0 nero	8 marrone
1 blu	9 bronzo
2 verde	10 verde chiaro
3 azzurro	11 azzurro chiaro
4 rosso	12 salmone
5 cremisi	13 porpora
6 giallo	14 arancio
7 bianco	15 grigio

Possiamo usare piu' facilmente questi 16 colori per "dipingere" i nostri grafici.

Ci sono due modalita' di variazione dei colori: si puo' far variare l'intero sfondo dello schermo (background color) oppure il colore con cui scrive la tartaruga (pen color).

### **Cambiamo lo sfondo**

**SETBG** (SETBACKGROUND)

```
SETBG listaRGB           oppure
SETBG unnumeroda0a15
```

lo schermo assume il colore corrispondente.

BG

e' un operazione che ha come output il numero o la lista corrispondente al colore dello schermo.

Esercizio:

facciamo scorrere sullo schermo 7 colori. Per visualizzare bene la situazione occorre introdurre una pausa tra una schermata e l'altra. Utilizzeremo di nuovo il comando WAIT numero (il numero indica le battute che il LOGO aspetta; un secondo contiene 60 battute).

TO COLORASCHERMO

```
SETBG 0 WAIT 20
SETBG 1 WAIT 20
SETBG 2 WAIT 20
SETBG 3 WAIT 50           rallenta il lampeggio
SETBG 4 WAIT 50
SETBG 5 WAIT 100          rallenta ancora
SETBG 6 WAIT 100
SETBG 7 WAIT 100
END
```

### **Facciamo disegnare colorato alla tartaruga**

**SETPC** (SETPENCOLOR)

```
SETPC listaRGB           oppure
SETPC unnumeroda0a15
```

questo comando fa usare alla tartaruga il colore corrispondente

Es: voglio disegnare in verde su uno sfondo giallo

```
SETBG 6 SETPC 2
```

### **PENCOLOR**

e' una operazione che ha come output il numero del colore della penna in uso.

### **Salviamo un disegno:**

Come le procedure, si possono salvare anche i disegno già fatti usando il comando SAVEPICT

```
SAVEPICT "nomefile
```

che vengono visualizzati nuovamente con

```
LOADPICT "nomefile
```

(attenzione: LOADPICT elimina i disegno già visualizzati, come un comando cs, e fa vedere poi quello caricato dal file, ma lascia la tartaruga nel punto e nella direzione precedente).

### **I salvataggi**

***Alla fine del lavoro, se hai lasciato che UCBlago salvi i file nella cartella che ha deciso lui (Documenti), devi***

***copiare i file su dischetto o pendrive su Windows:***

- apri Esplora Risorse, e vai nella cartella Documenti. Questa non sta sempre nello stesso posto, ma e' abbastanza facile trovarla
- seleziona i file che ti interessano e copiali (tasto destro -> copia)
- vai ora su "Risorse del Computer". Individuare l' unita' dischetto (di solito e' la prima e si chiama "Floppy da 3,5

pollici (A:))" oppure l' unità pen-drive (di solito e' l' ultima e si chiama "Disco rimovibile (X:)": X è una lettera scelta dal sistema). Apri l' unita' e, se hai una cartella particolare dove copiare, apri anche quella.

- incolla (tasto destro -> incolla).

***se questo sistema non ti piace, la prossima volta puoi***

**istruire subito UCBlogo a salvare su dischetto o pendrive**

- Vai su "Risorse del Computer". Individuare l' unita' dischetto (di solito e' la prima e si chiama "Floppy da 3,5 pollici (A:))" oppure l' unità pen-drive (di solito e' l' ultima e si chiama "Disco rimovibile (X:)": X è una lettera scelta dal sistema). Apri l' unita' e, se hai una cartella particolare dove copiare, apri anche quella.
- Nella barra degli indirizzi di Esplora risorse vedrai il cammino completo che ti porta nella cartella dove vuoi salvare.
- in UCBlogo, prima di salvare qualsiasi cosa, devi dare il comando

**SETPREFIX** "ilcamminochevedi

per esempio: SETPREFIX "G:\salvalogo

- verifica di averlo fatto giusto dando il comando

**PRINT PREFIX**

che pero' ti dara' una sorpresa: la barra rovescia "\" non c'e' piu'.

Questo perche' per il Logo la barra rovescia e' un carattere speciale, che dice al Logo "prendi il carattere che segue come sta, non dargli un significato sintattico anche se e' uno spazio, un doppio punto o le doppie virgolette". E per dare al Logo una barra rovescia bisogna dirgli che le deve prendere come sta, facendola precedere... da un' altra barra rovescia.

In questo caso non fa niente: G:salvalogo viene capito comunque da Windows. Ma se tu volessi usare una cartella piu' interna, per esempio

SETPREFIX "G:\luigi\scuola\salsalvalogo

dovresti proprio scrivere esplicitamente

```
SETPREFIX "G:\\luigi\\scuola\\salvalogo
```

### **Come rallentare la tartaruga:**

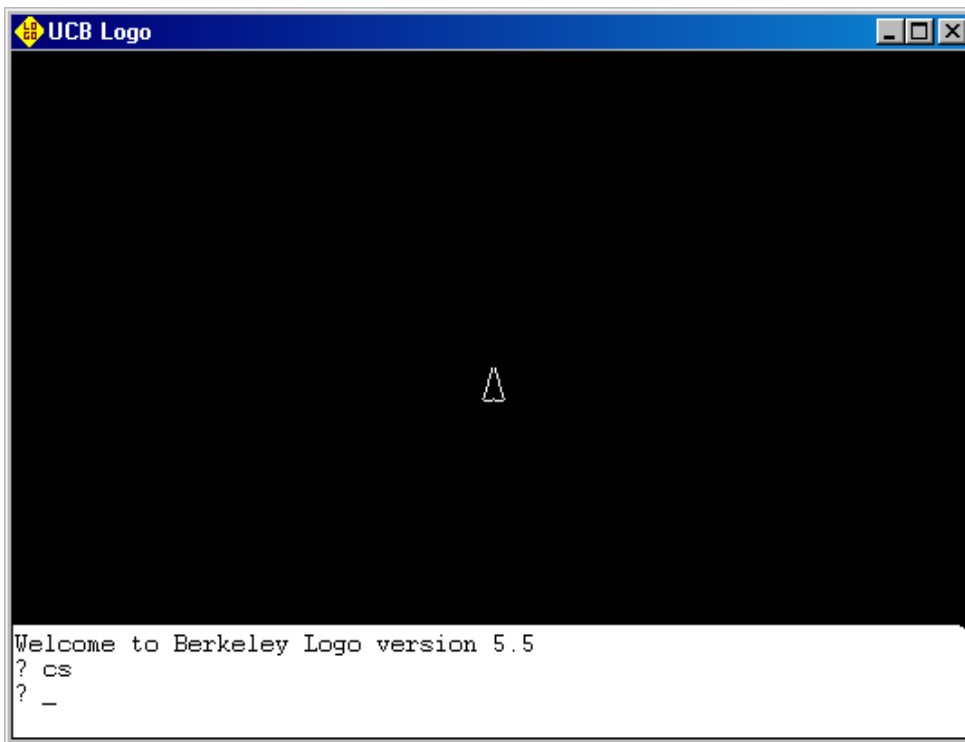
La tartaruga traccia sullo schermo i disegni molto rapidamente, tanto che tu non riesci a vedere quello che fa e alle volte non riesci a capire dove hai sbagliato perche' vedi solo il risultato finale. Quindi puo' risultare utile rallentare la tartaruga in modo che esegua passo passo la procedura che le hai insegnato. Per far questo dai il comando

### **step procedures**

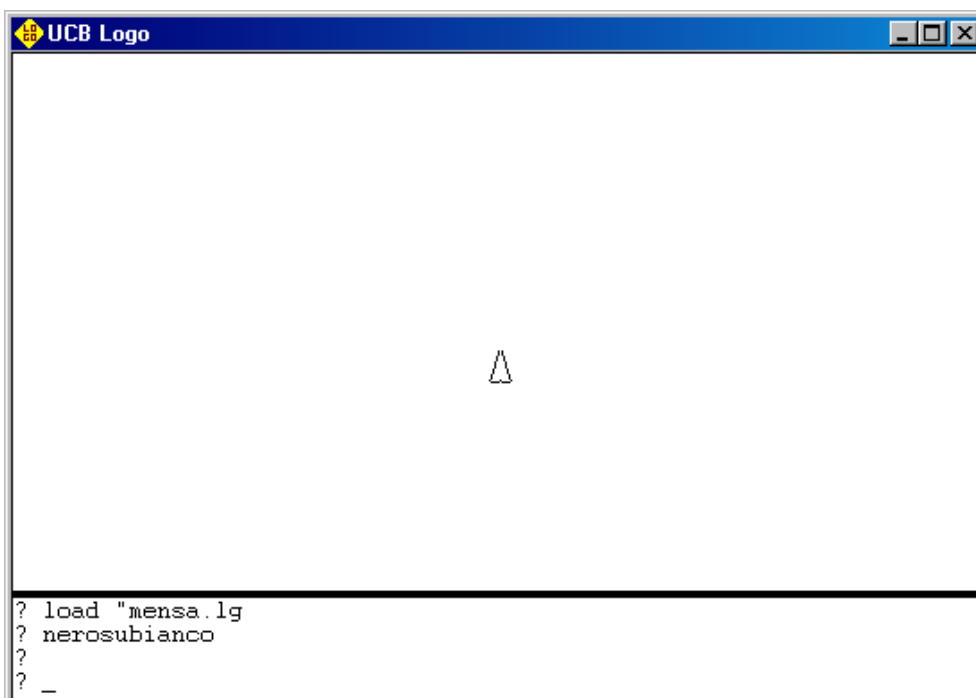
da questo momento in poi per far avanzare la tartaruga lungo il percorso che le hai insegnato, dopo aver chiamato la procedura, devi premere ripetutamente il tasto invio, e la tartaruga fara' un tratto ad ogni invio. quando vuoi tornare alla situazione normale dai il comando inverso

### **unstep procedures**

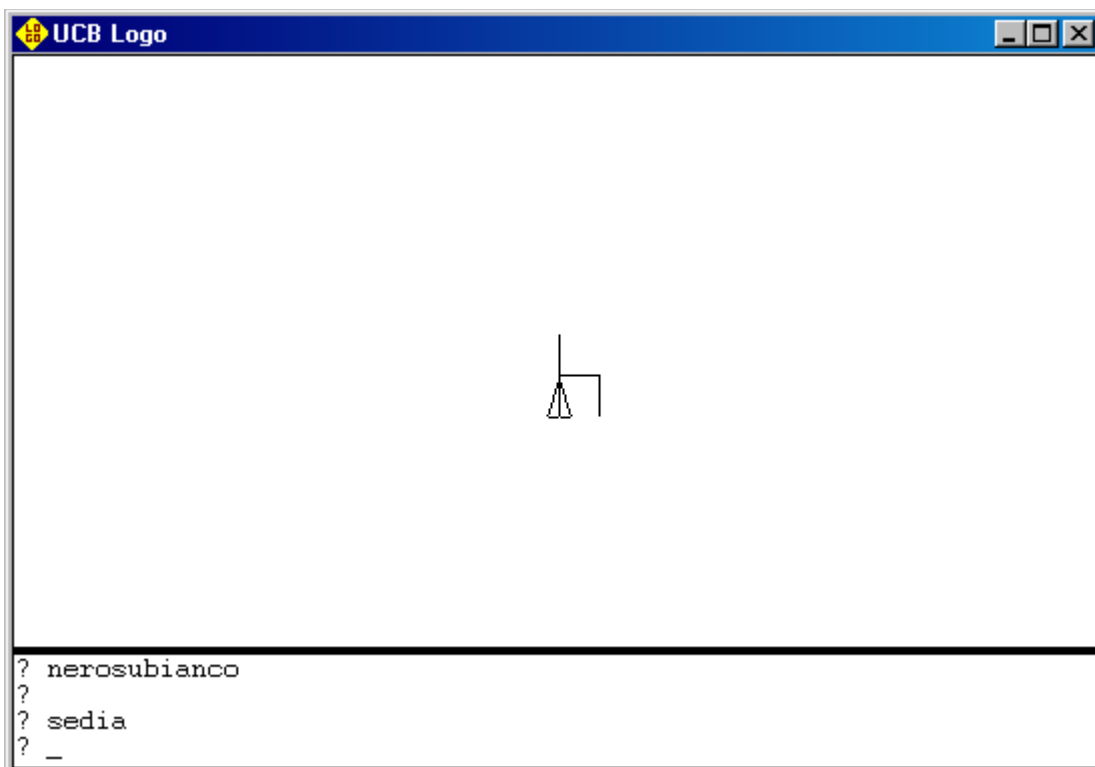
## Esempi di procedure grafiche



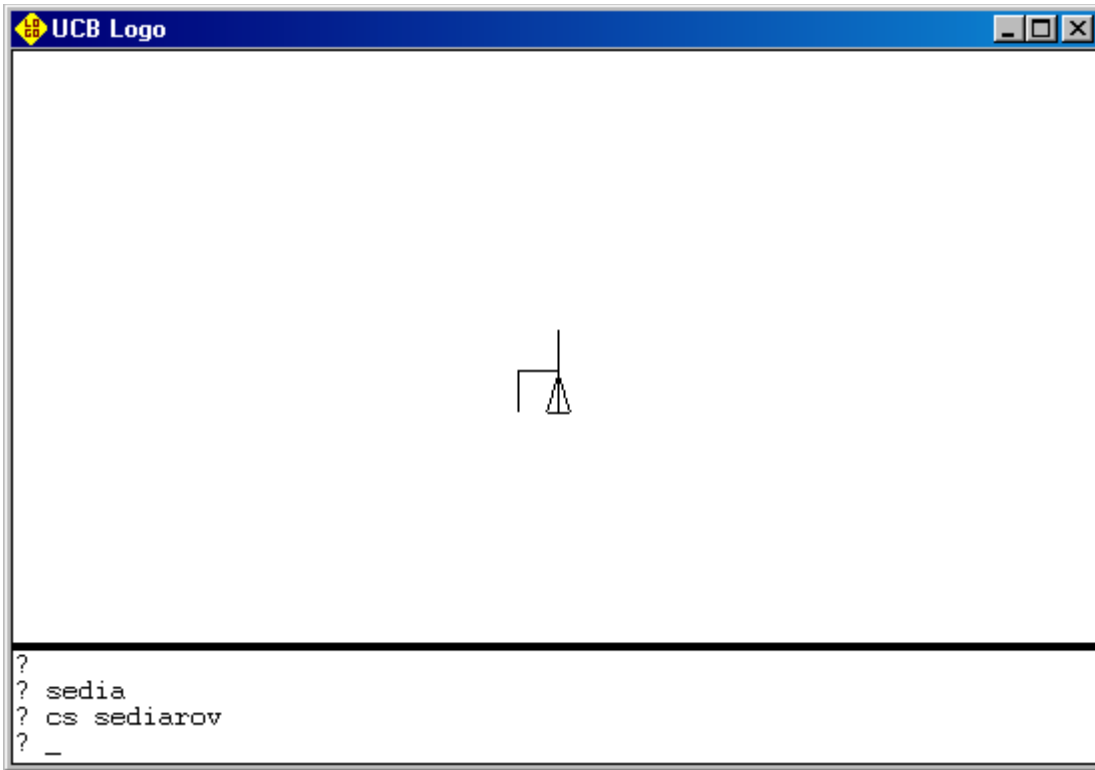
```
to nerosubianco  
  cs  
  setbg 7  
  setpc 0  
end
```



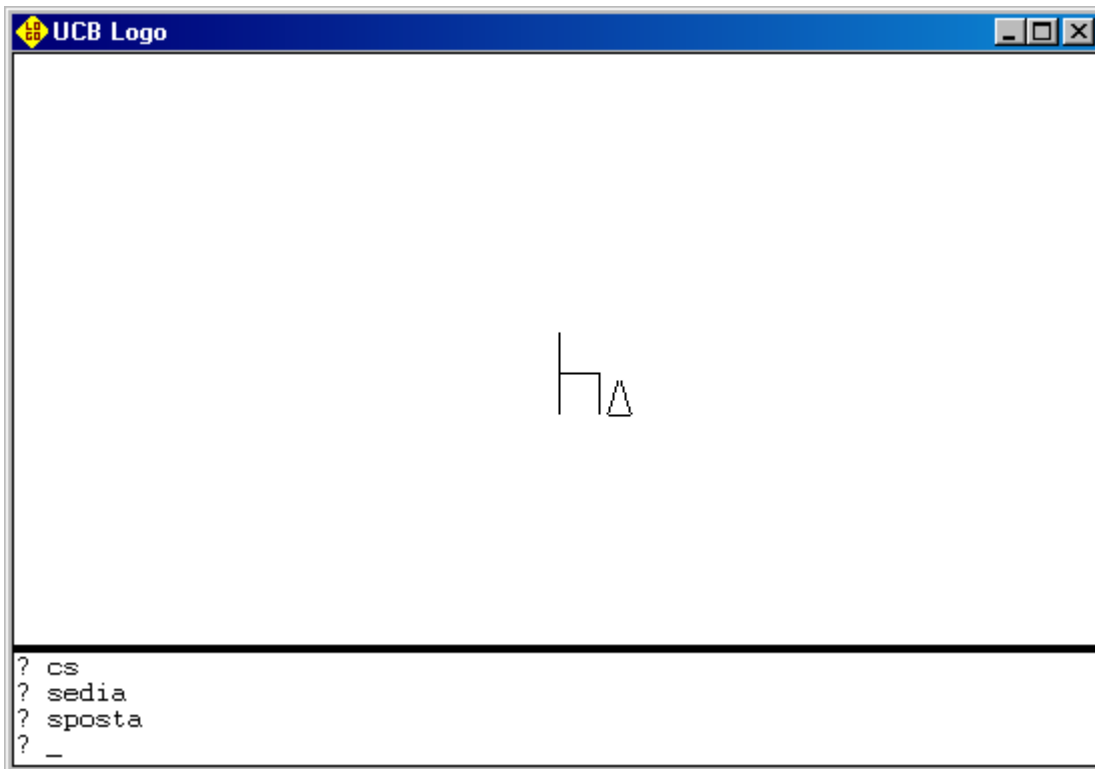
```
to sedia
fd 40 bk 20
rt 90 fd 20
rt 90 fd 20
lt 180
fd 20 lt 90
fd 20 lt 90
fd 20 rt 180
end
```



```
to sediarov
fd 40 bk 20
lt 90 fd 20
lt 90 fd 20
rt 180
fd 20 rt 90
fd 20 rt 90
fd 20 lt 180
end
```

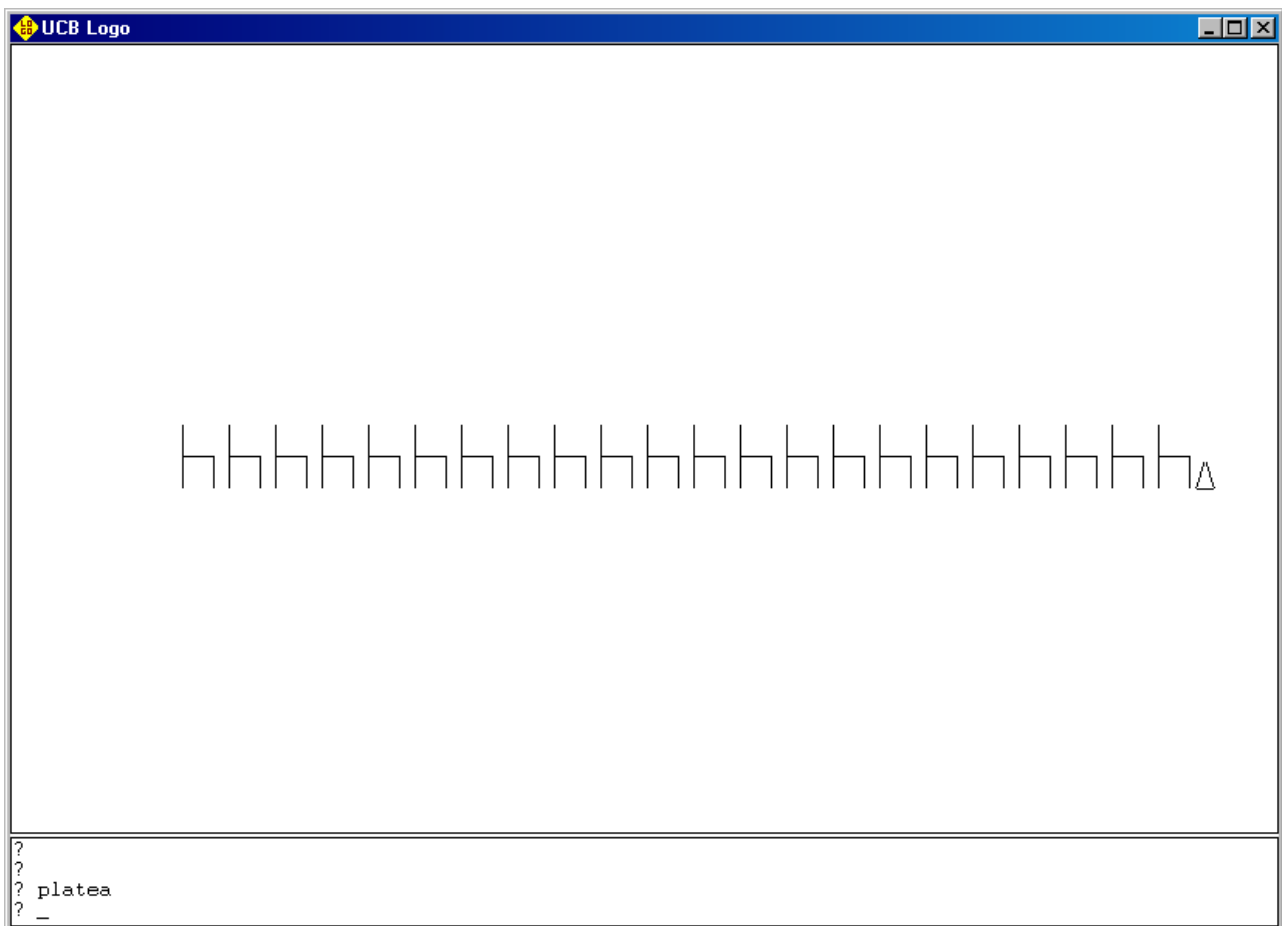


```
to sposta  
  pu  
  rt 90 fd 30 lt 90  
  pd  
end
```

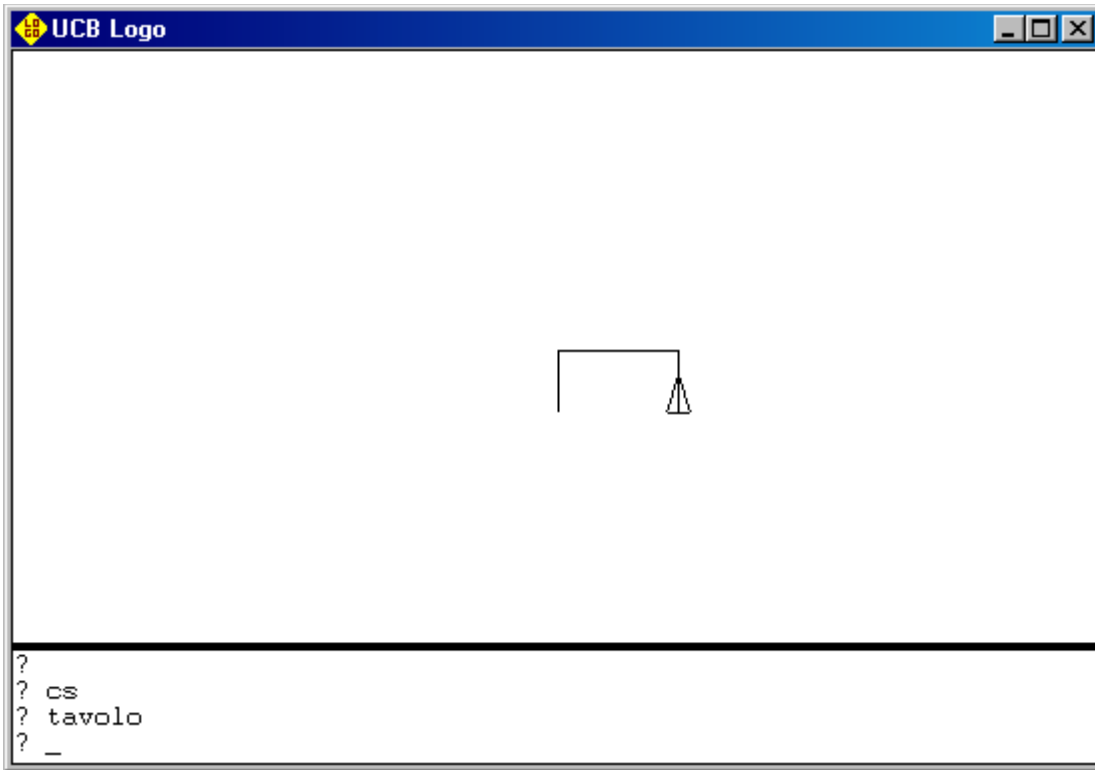


```
to inizio
pu
home
lt 90 fd 200 rt 90
pd
end

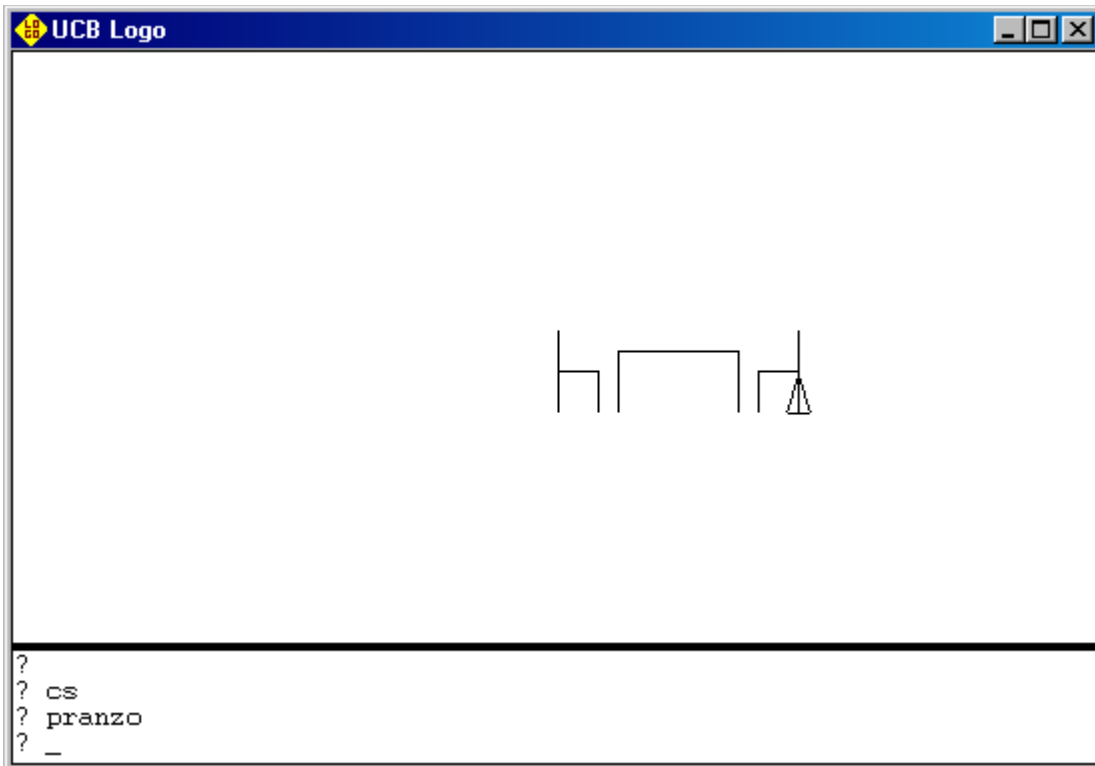
to platea
inizio
repeat 22[sedia sposta]
end
```



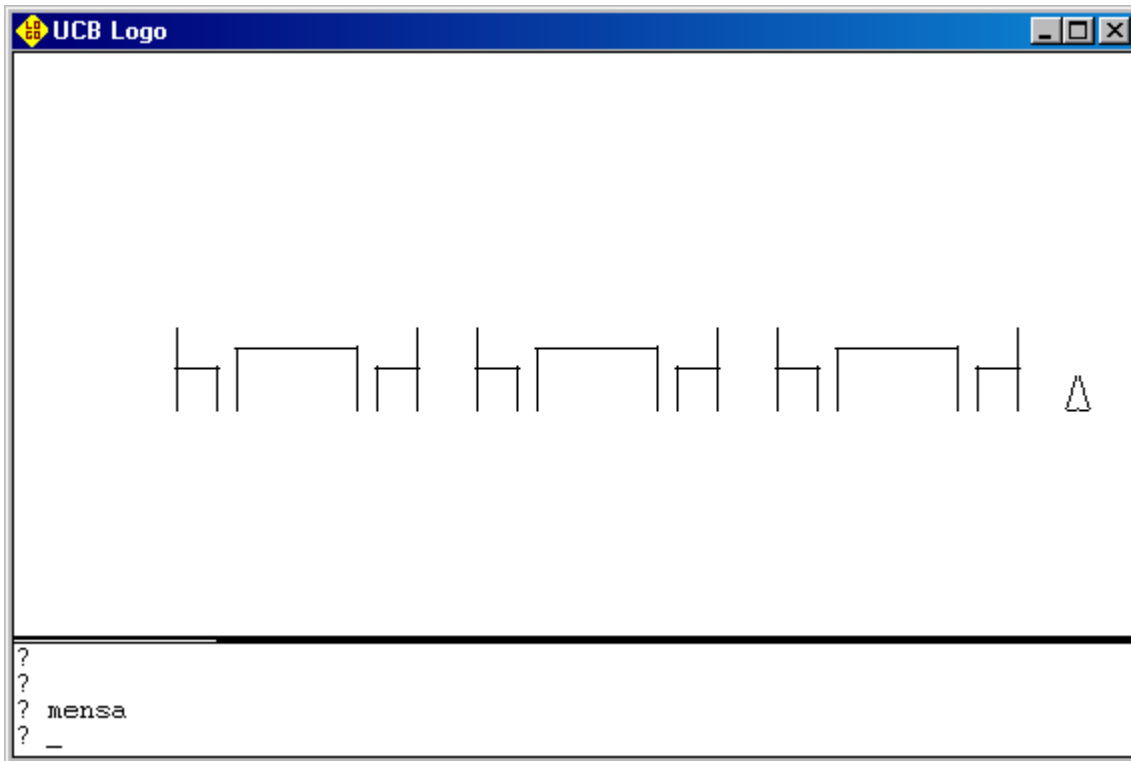
```
to tavolo
fd 30 rt 90
fd 60 rt 90
fd 30 rt 180
end
```



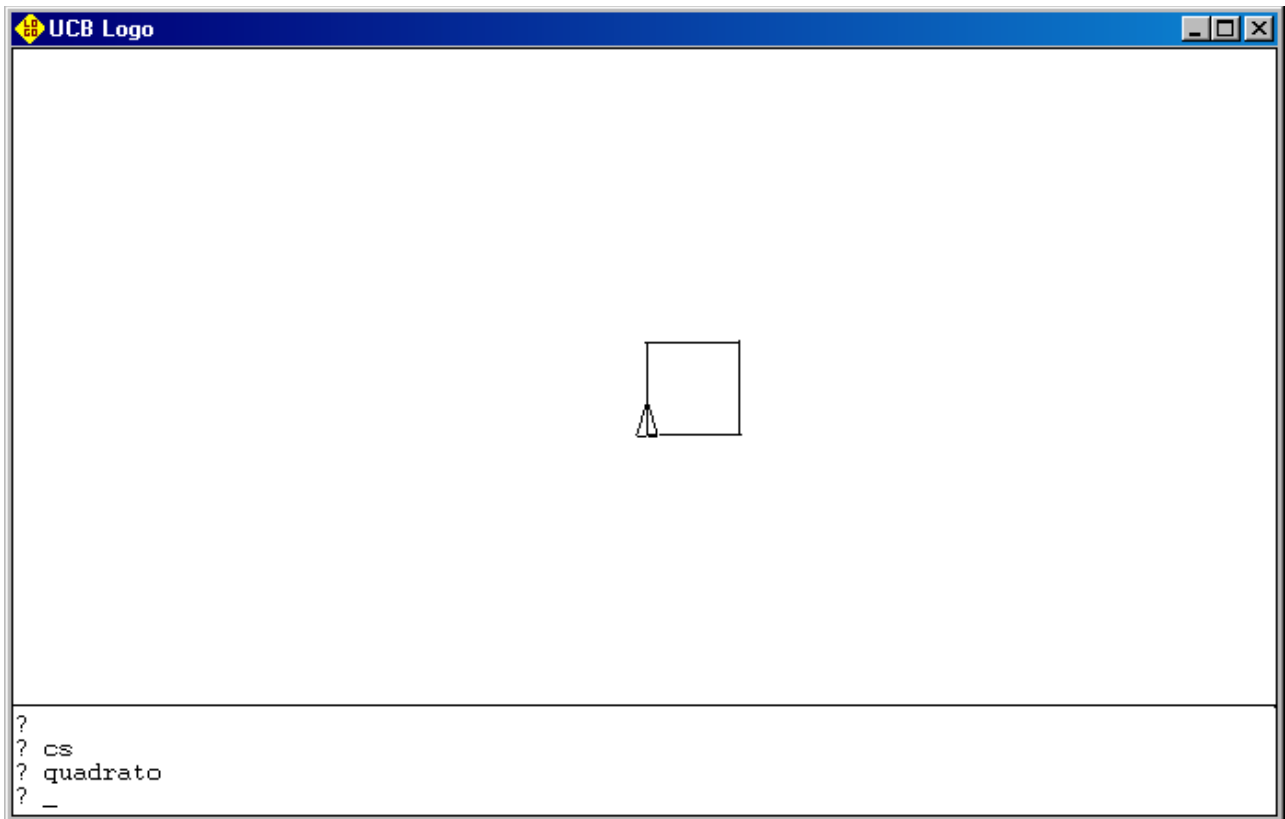
```
to pranzo  
sedia sposta tavolo sposta sediarov  
end
```



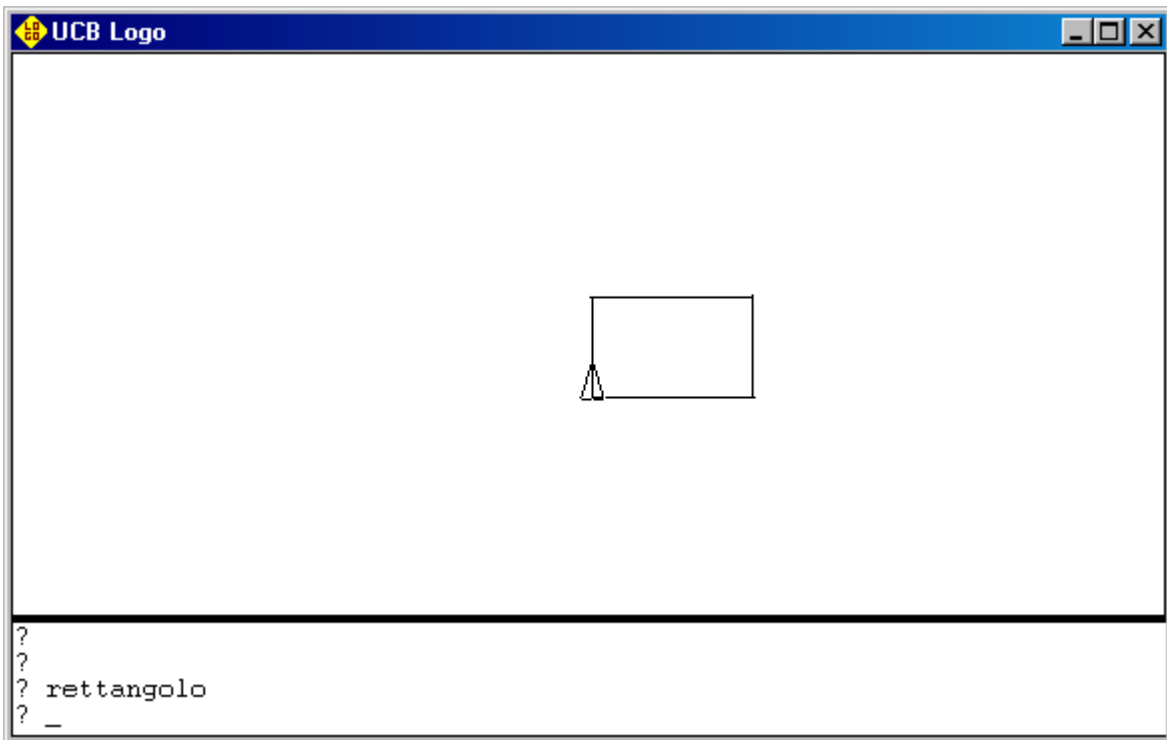
```
to mensa
inizio
repeat 3[pranzo sposta]
end
```



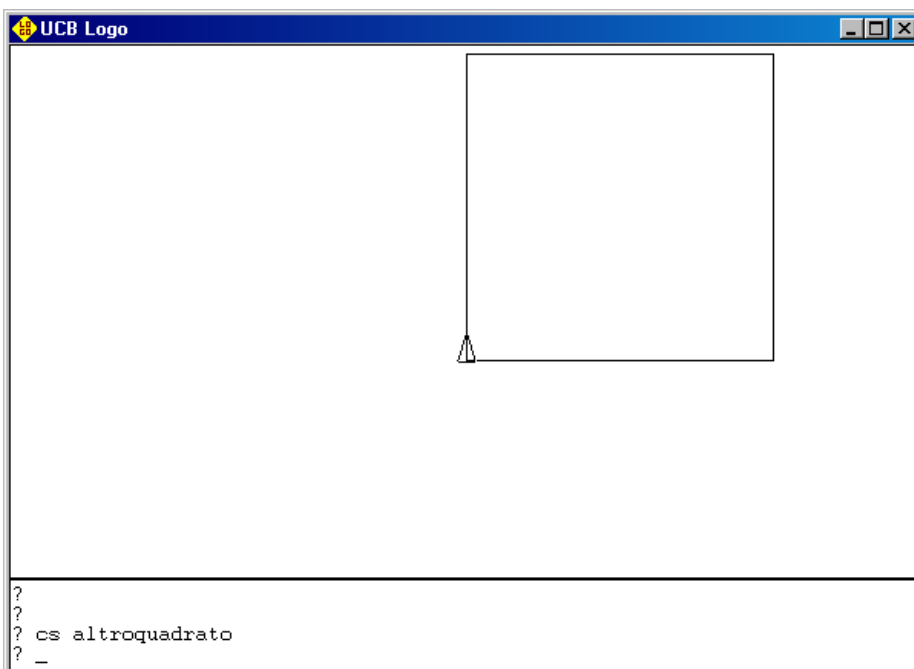
```
TO QUADRATO
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
END
```



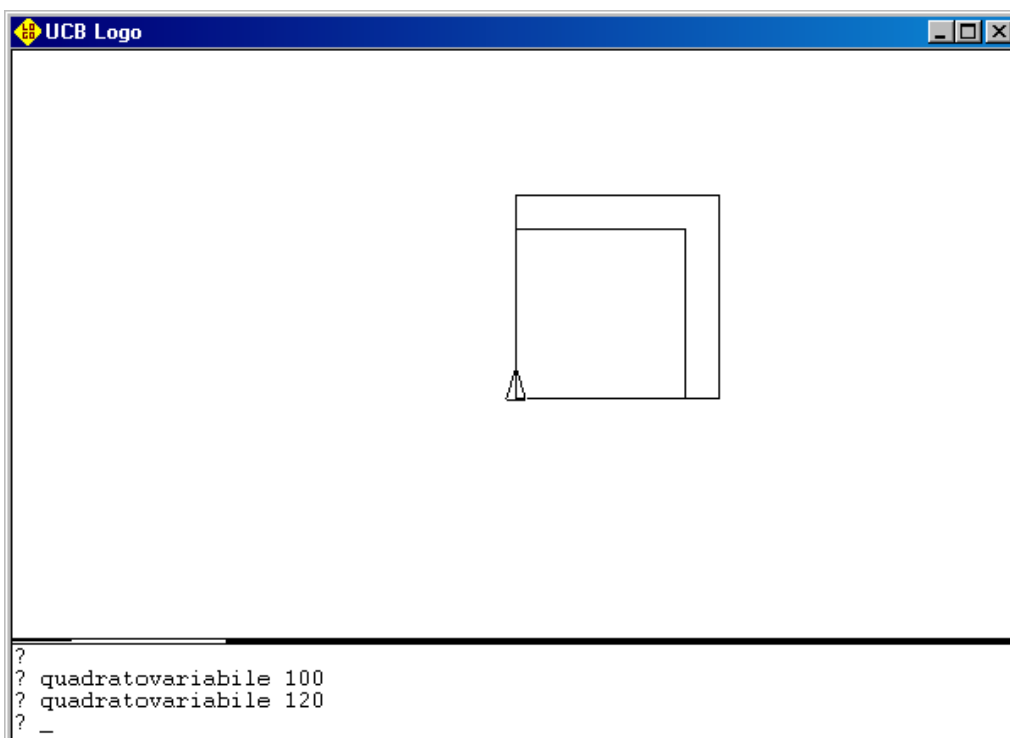
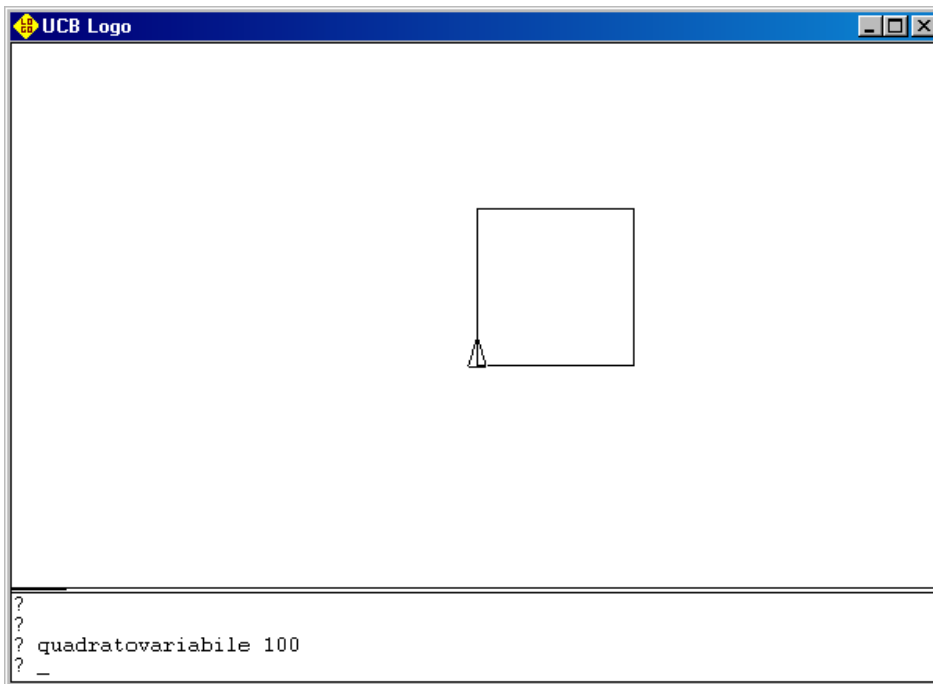
```
to rettangolo  
  fd 50 rt 90  
  fd 80 rt 90  
  fd 50 rt 90  
  fd 80 rt 90  
end
```



```
to altroquadrato  
  repeat 4 [fd 200 rt 90]  
end
```



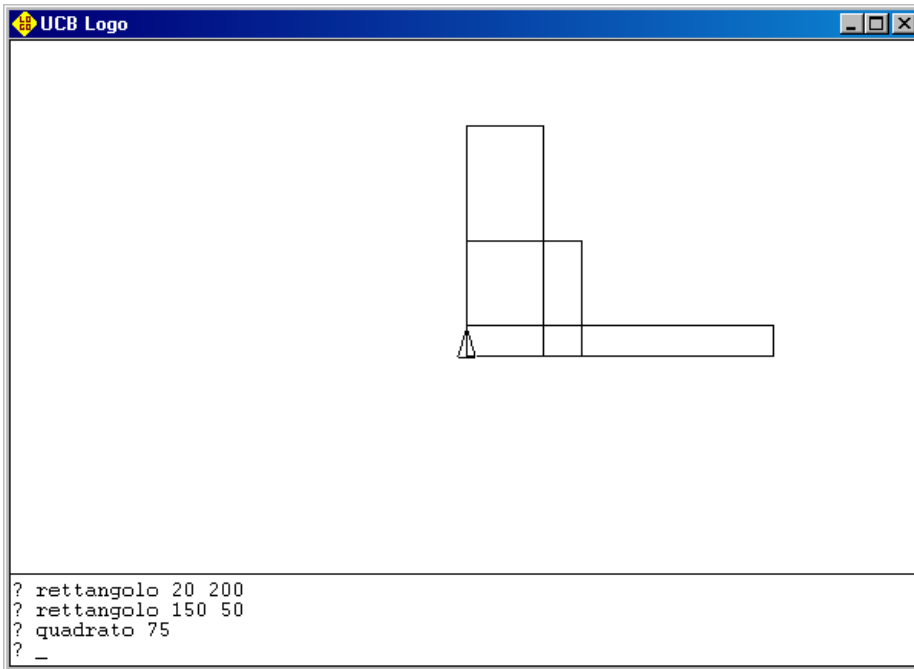
```
to quadratovariabile :lato  
  repeat 4 [fd :lato rt 90]  
end
```



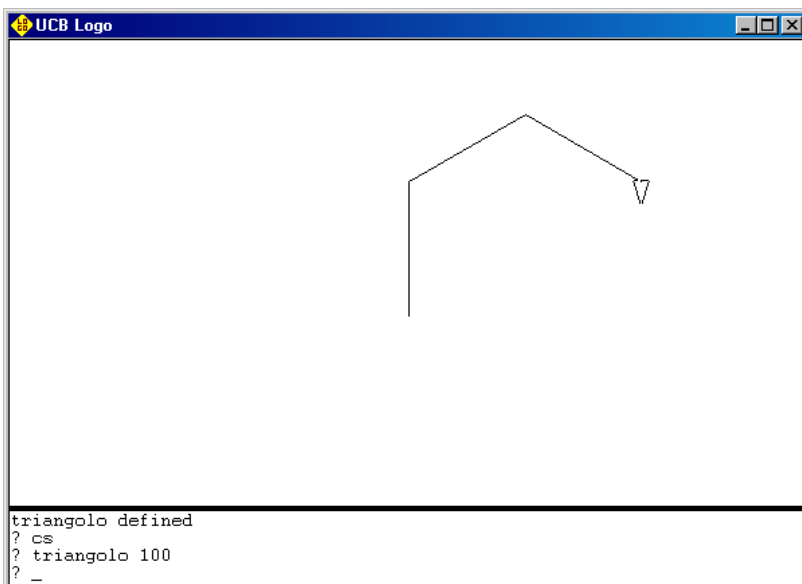
```
to rettangolo :lato1 :lato2  
  repeat 2 [fd :lato1 rt 90 fd :lato2 rt 90]  
end
```

```
erase "quadrato

to quadrato :lato
rettangolo :lato :lato
end
```



```
to triangolo :lato
fd lato rt 60
fd lato rt 60
fd lato rt 60
end
```

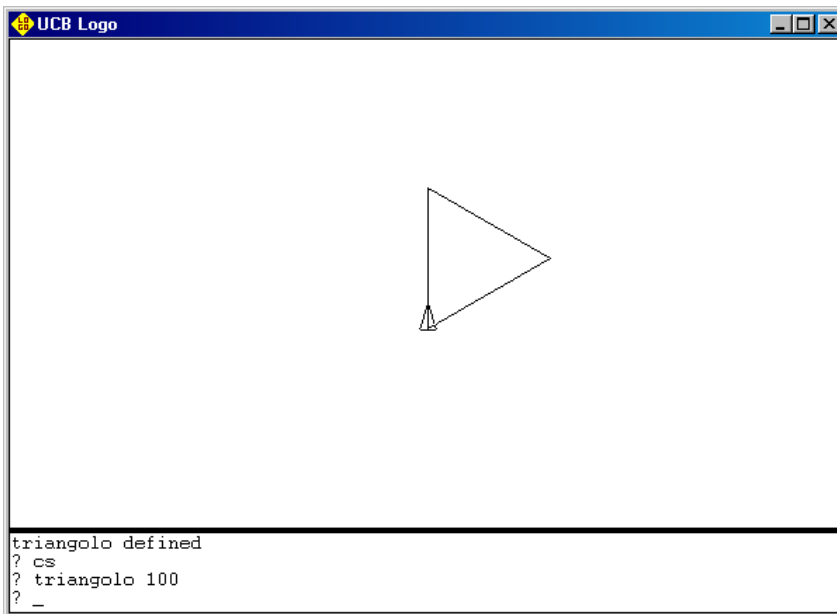


**Un bug inatteso?? Questo non è un triangolo!**

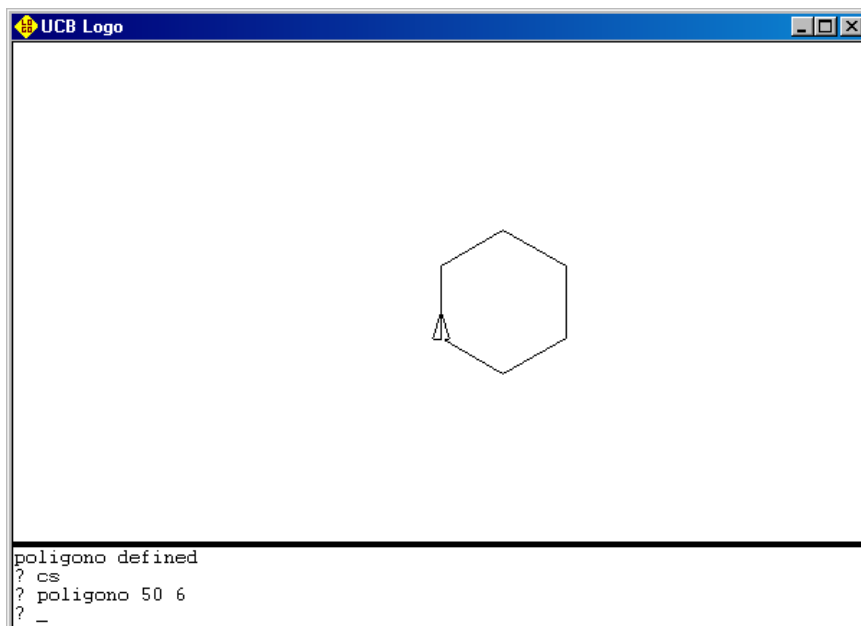
**Il teorema del giro completo della tartaruga:**

```
erase "triangolo
```

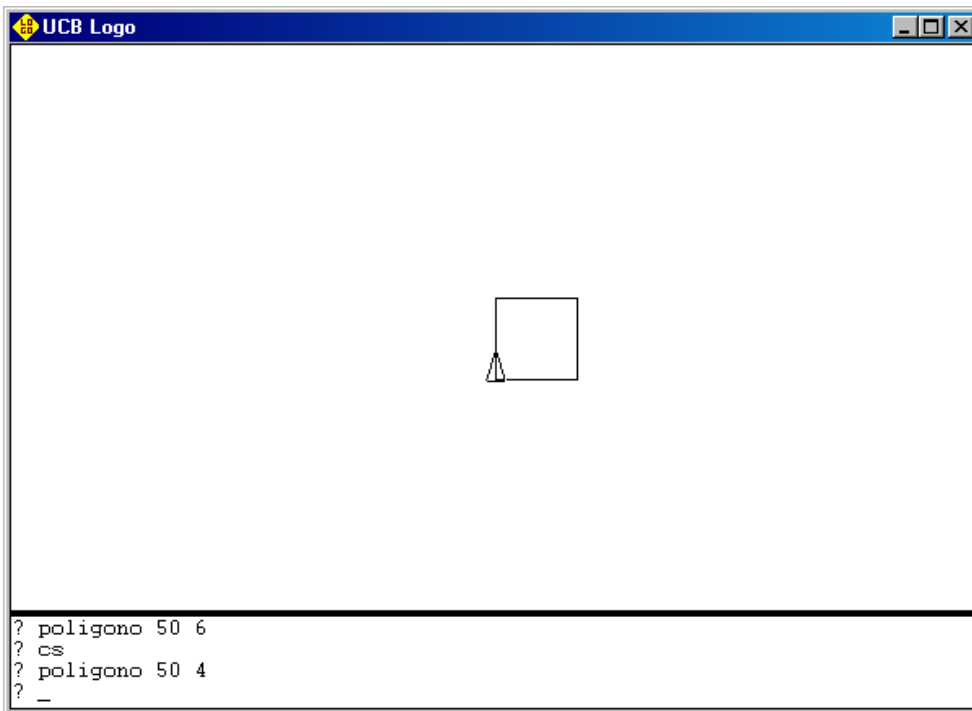
```
to triangolo :lato  
repeat 3 [fd :lato rt 360/3]  
end
```



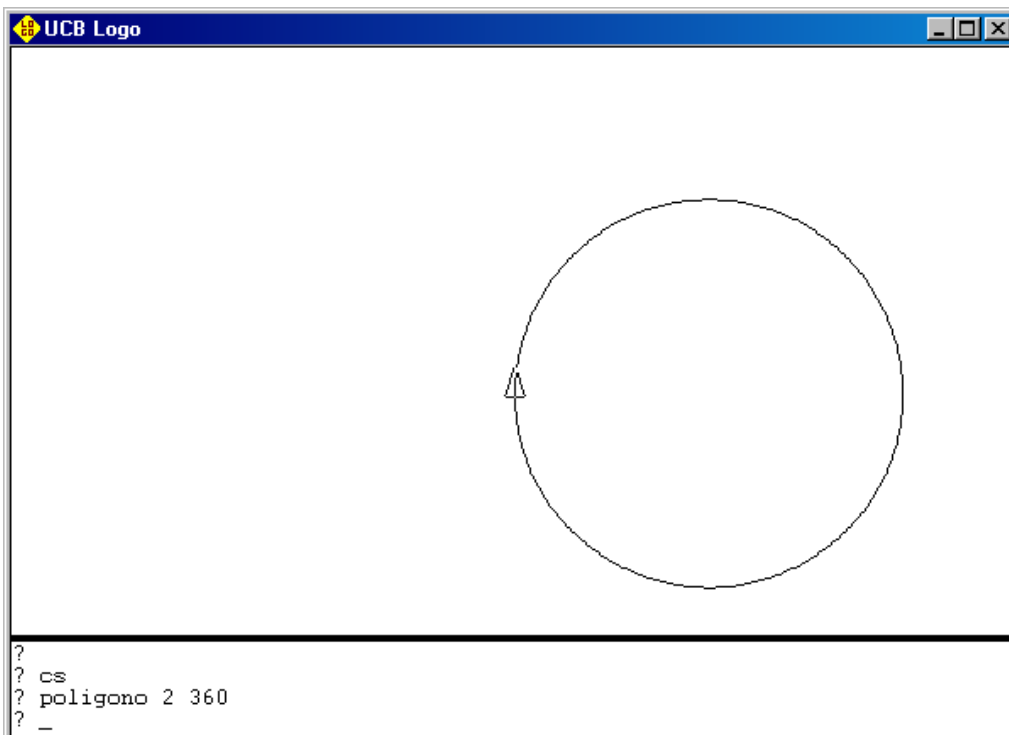
```
to poligono :lato :lati  
repeat :lati [fd :lato rt 360/:lati]  
end
```



un altro modo di fare il quadrato:

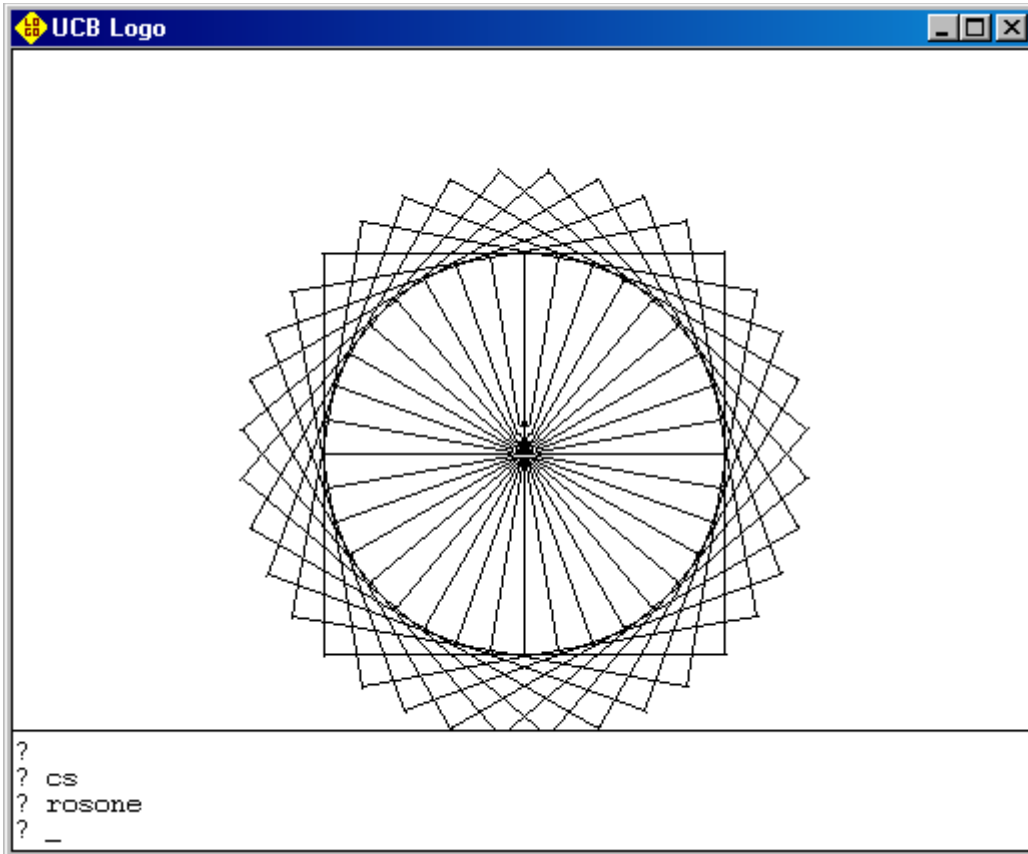


quasi un cerchio:

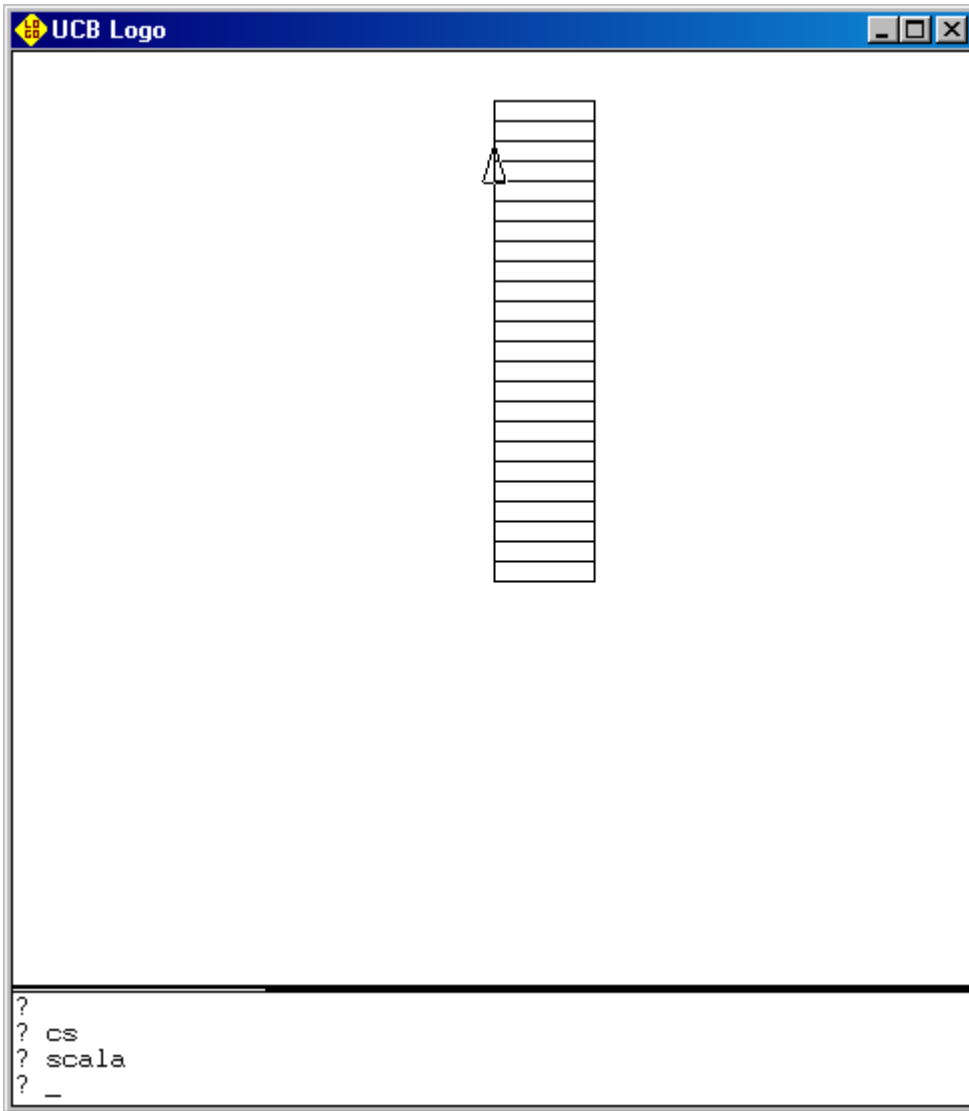


## divertimenti

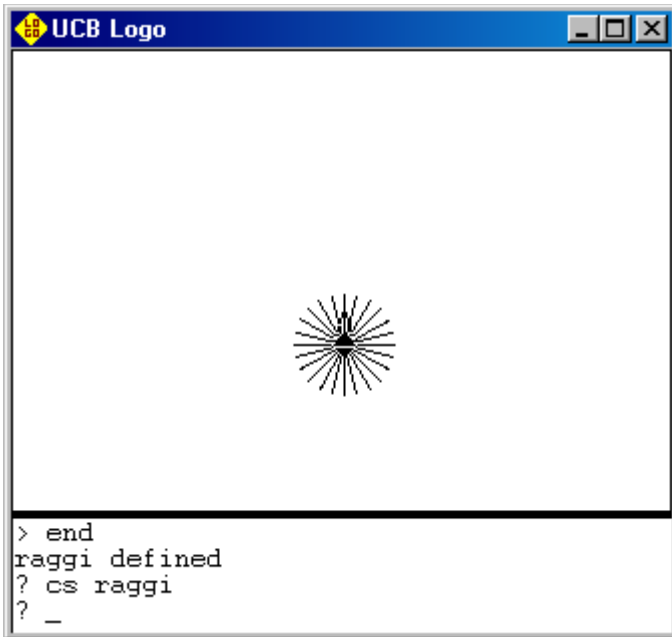
```
to rosone  
repeat 36 [quadrato 100 rt 10]  
end
```



```
to scala  
repeat 20 [quadrato 50 a 10]  
end
```



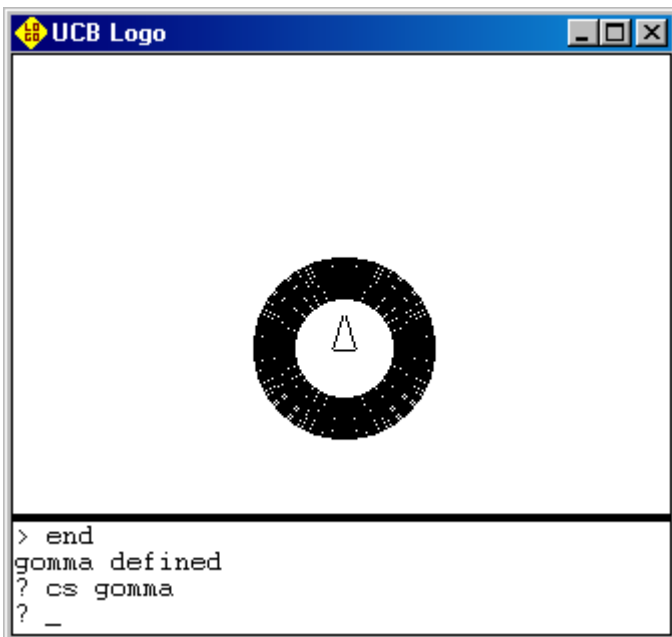
```
to raggi
repeat 24 [fd 25 bk 25 rt 15]
end
```



```

to gomma
  pu repeat 360 [fd 25 pd fd 20 pu bk 45 rt 1]
  pd
end

```



questa procedura funzionava bene sotto il dos con schermo a bassissima risoluzione, con la alte risoluzioni occorre un tratteggio molto più fine:

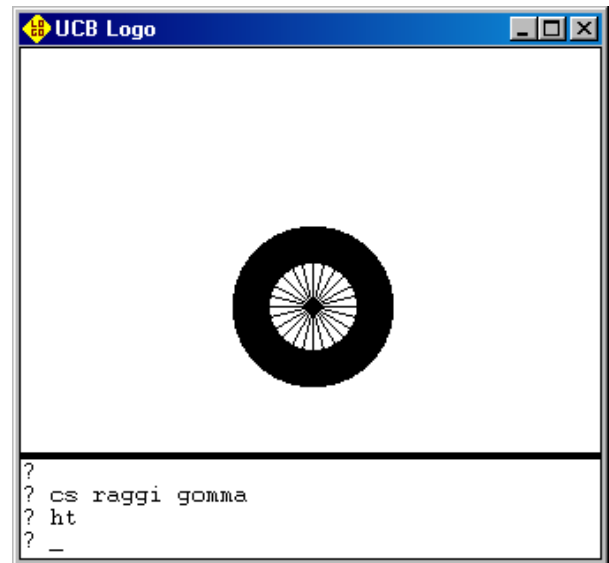
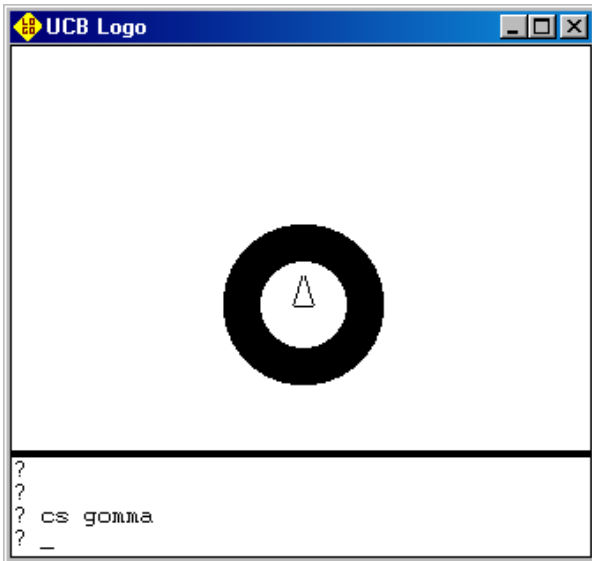
```
erase "gomma
```

```

to gomma
  pu repeat 36000 [fd 25 pd fd 20 pu bk 45 rt 0.01] pd
end

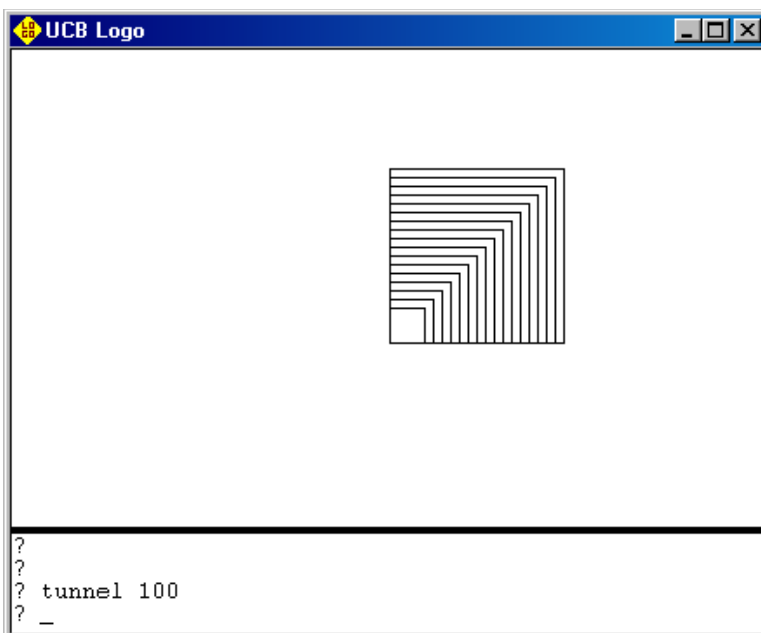
```

(che però ci mette un po' di tempo... esistono altri modi.)

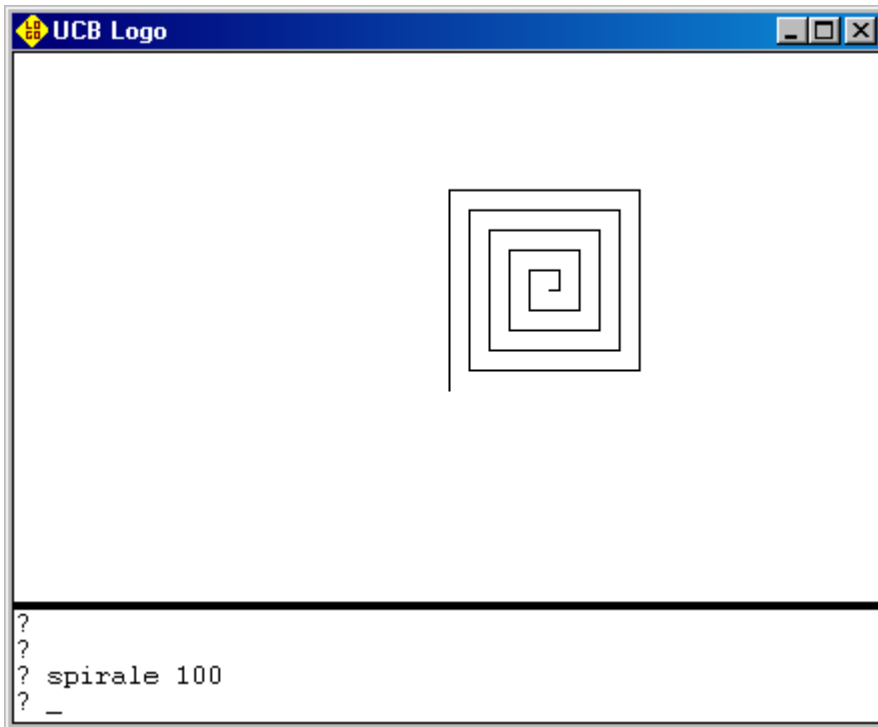


### Ricorsività

```
to tunnel :lato  
  if :lato < 20 [stop]  
  quadrato :lato  
  tunnel :lato -5  
end
```

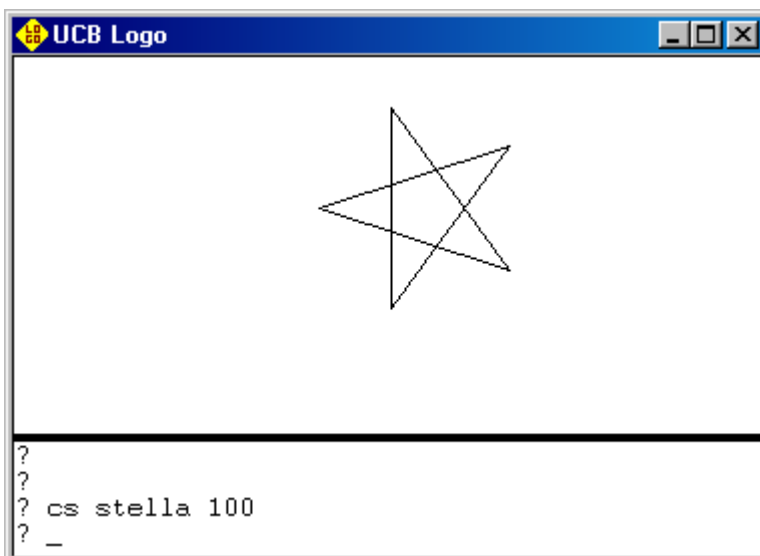


```
to spirale :lato
if :lato < 0 [stop]
fd :lato rt 90
spirale :lato - 5
end
```



### A caso

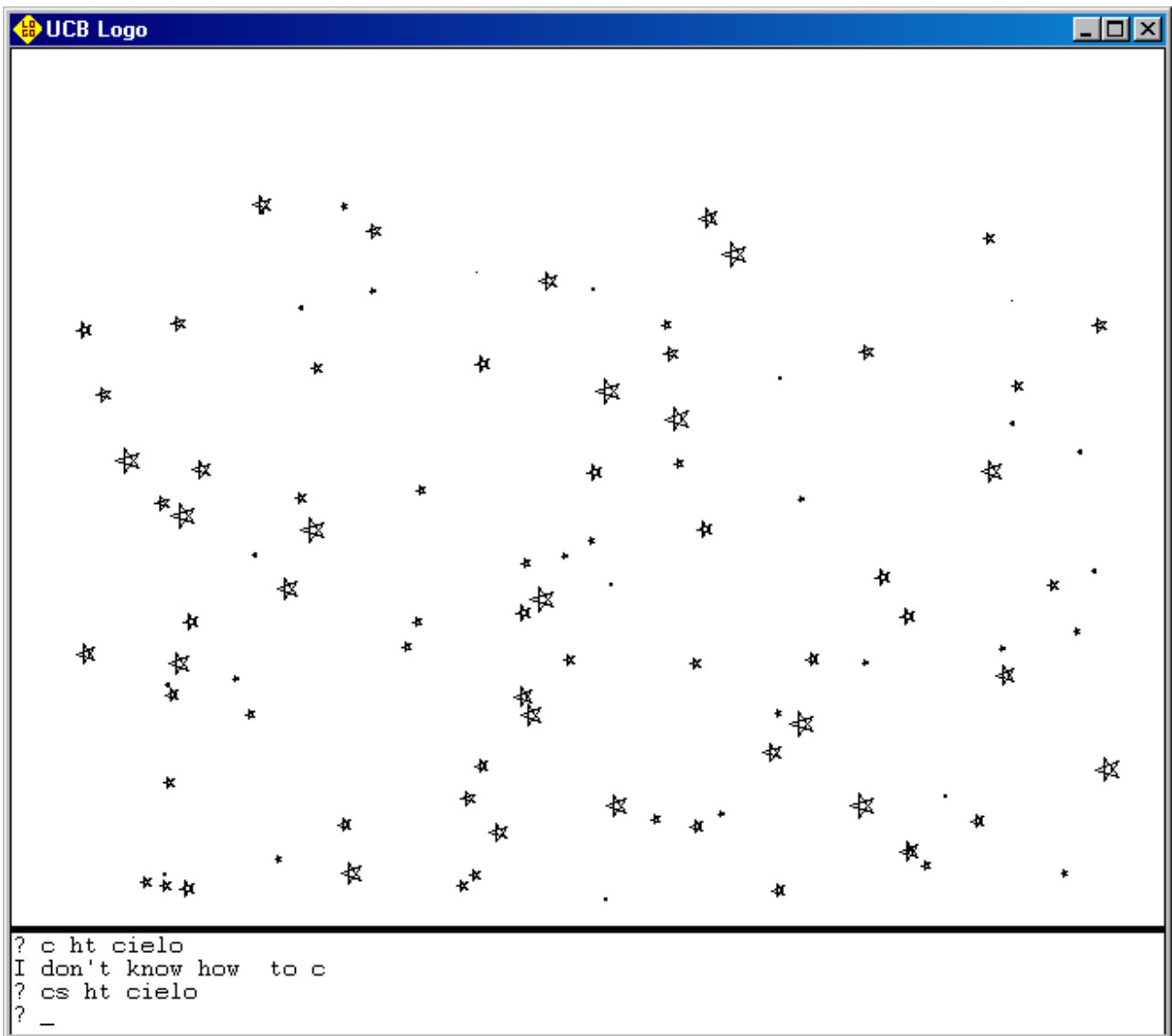
```
to stella :misuralato
repeat 5 [fd :misuralato rt 360*2/5]
end
```



```
to spostax
setx (-300 + random 600)
end

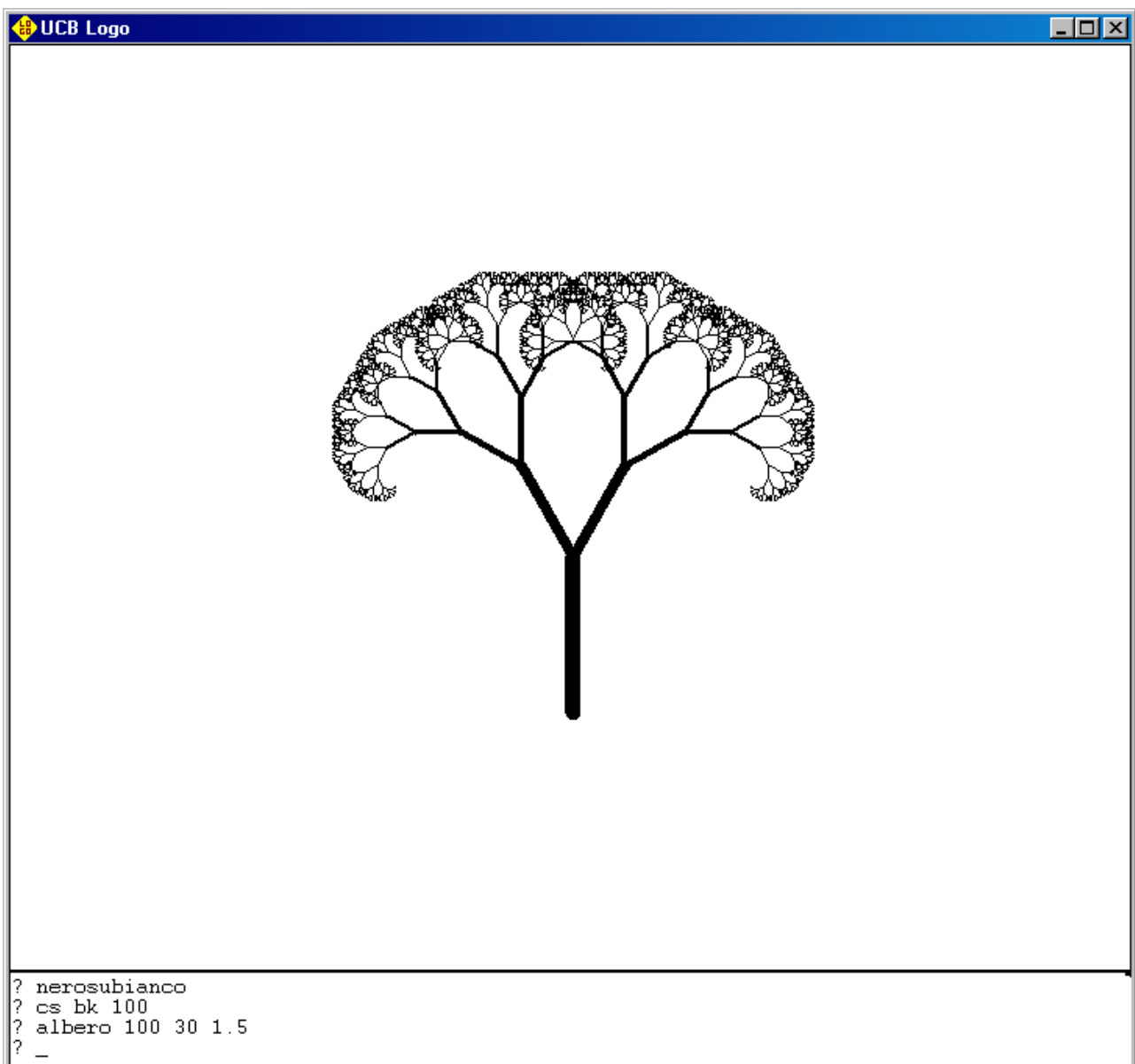
to spostay
sety (-200 + random 400)
end

to cielo
repeat 100 [stella random 15 pu spostax spostay pd]
end
```



## Ricorsività spinta

```
to albero :ramo :angolo :rapporto
if :ramo < 1 [stop]
setpensize round (:ramo/10)
fd :ramo
lt :angolo
albero :ramo/:rapporto :angolo :rapporto
rt 2*:angolo
albero :ramo/:rapporto :angolo :rapporto
lt :angolo
bk :ramo
end
```



- o -